

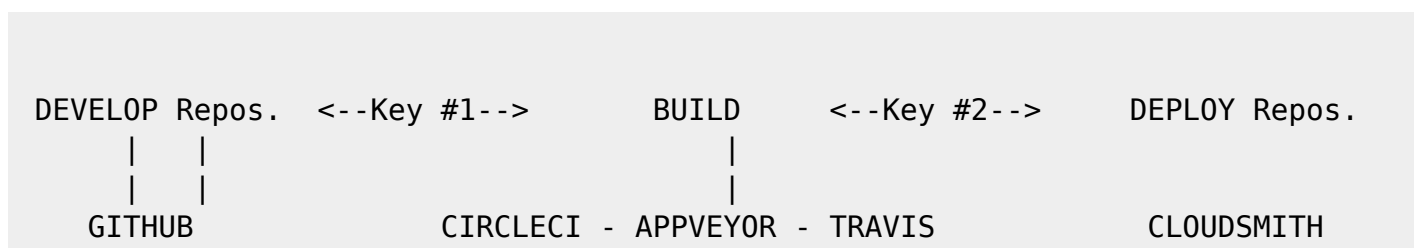
# PI Manager Dev Setup

[Plugin Manager Github Wiki](#) Extensive documentation

## Build and Deploy Plugins

Alec Leamas is the author of PI Installer and should be given extensive credit & thanks for his many months of work on the Plugin Installer. Jon Gough has also generously spent many months to improve the PI Dev experience by his extensive work revising and improving the Build and Deploy side of PI Installer, the CMake and CI files needed at the head of any plugin that supports PI Installer.

To understand more detail about the entire system, please also review [Plugin Installer Documents](#)



NOTE:

- It is important to get Key #1 and Key #2 entered correctly, provided you need them.
- KEY #1 is only needed to PUSH the build back to your Github RELEASE Repository.
- Github OpenSource accounts are Public and easily accessible for download by the BUILD services.
- KEY #2 is needed to give the BUILD services the ability to PUSH to CLOUDSMITH.

## PI Manager System consists of three parts:

### 1. Plugin Manager inside OpenCPN

The plugin manager is now in OpenCPN master branch. Make a fresh build of opencpn from this branch. There are occasional improvements made now.

### 2. Plugin Build & Deploy by Plugin Devs

 **Fix Me!** We need to simplify

1. Step by step
2. List of files needed
3. Provide two sample file packages for download

#### 4. To be applied to the plugin (provided they are working)>

CMake and CI Files from Jon Gough's much improved [testplugin\\_pi](#) using the “cmake\_flatpak\_test” branch. These are pre-configured with the code required to create the compressed binary and metadata xml files needed for the PI Installer to work. These files create plugin binary and metadata xml catalog files for various operating systems, utilizing git, the plugin's GitHub repository, Appveyor, Travis and Circleci to build, and the Cloudsmith repository for deployments. The earlier [squiddio\\_pi](#) CMake and CI Files were a significant step in the development process, however we recommend use of the files in Testplugin\_pi. Also please note that the CI parts, while useful, are not necessary to use the installer. As long as there is a working file with metadata and a downloadable tarball everything is fine, at least for a starter.

### 3. Metadata Repository:Git OpenCPN/Plugins



1. Diagram: to explain relationship between CLOUDSMITH REPO and OPENCN/PLUGINS
2. Purpose: To collect all the plugins metadata.xml's for the tarballs and push them to opencpn/plugins master:beta:alpha branches.

[OpenCPN plugins project README](#) The github repository that stores plugin metadata files (submitted by Pull Request), and the python code to create a single “ocpn-plugin.xml” file of plugin metadata for use by the PI Installer. This file contains all the necessary PI information and locations of files. Dave R. has control of this github repository and will accept PRs

### Assumed Cloud Services

We assume you have or are able to configure the following free opensource accounts/services:

1. Code Repository [GitHub](#)
2. Deployment Repository [Cloudsmith](#)
3. Build Tool [CircleCI](#) Needs authorizations from Code & Deploy repositories. Preferred.
4. Build Tool [Appveyor](#) Needs authorizations from Code & Deploy repositories.
5. Build Tool [Travis-CI.org](#) Needs authorizations from Code & Deploy repositories.
6. Build Tool [Travis-CI.com](#) Needs authorizations from Code & Deploy repositories.

These next steps add build and deployment of multiple OS to Git Release and Cloudsmith for PI Installer.


### Plugins Adaptation

You may have to adapt the plugin in various ways. Please review the items in this link [Plugin Adaptation](#)

# GITHUB Personal Token Generation and Use

KEY #1 provides your BUILD (Appveyor, Travis-ci or Circleci) service the ability to PUSH artifacts (tarballs, xml, packages) back to your GitHub RELEASE repository.

Provide access for APPVEYOR, TRAVIS-CI and CIRCLECI to your plugin's GitHub repository, so that these

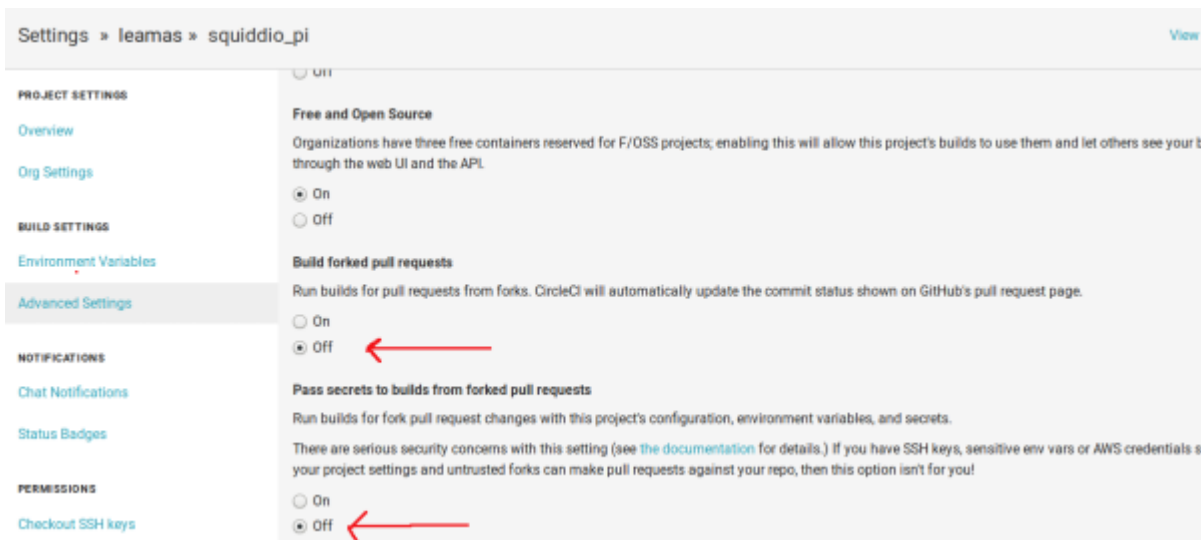
accounts can clone the repository in order to build the plugin.  (This needs to be confirmed, it is an opensource public account, so no auth\_key may be needed.)

1. From your GITHUB account [Username]
  1. Select Settings
  2. Pick Developer Settings
  3. Pick Personal Access Settings
  4. Enter 'git-ptoken-[plugin]-into-appveyor' or 'git-ptoken-[plugin]-into-circleci' or similar, etc
  5. Then copy and save the 'number' for use below to provide CIRCLECI, APPEVEYOR and TRAVIS with credentials to access and download from your personal GitHub repository (for each plugin and each service).
  6. You will need a separate Github Personal Access Token for each plugin, and each service (Appveyor, Travis)
2. From your CIRCLECI account
  1. See [CIRCLECI \(Build Account\)](#)
3. From your APPVEYOR account
  1. Plugin, then select Setting (gear on top line to the right)
  2. Pick Environment (left menu)
  3. Pick Environment variables (in the center entry area).
  4. Pick Add Variable.
  5. Enter name "GIT-PTOKEN-[pluginname]-INTO-APPVEYOR"
  6. Enter value, which you saved, from GitHub.
  7. Pick SAVE at the bottom.
  8. Now simply use "GIT-PTOKEN-[pluginname]-into-APPVEYOR" with your appveyor.yml file
  9. Note: Appveyor used to give you an encryption number to at the bottom of the appveyor.yml file [Appveyor.yml file example](#).
4. For TRAVIS-CI [Travis-CI Encryption Docs](#)
  1. First install [Travis CLI](#) to encrypt a github secure public key.
  2. Then [Encrypt a Github API Token with Travis CLI Encryption](#) using the Personal Access Token 'number' you saved from Git.
  3. Then use the resulting 'encryption' in the [.Travis.yml file example](#).
  4. For more information about Travis encryption of a Github Personal Access Token see [CI:Travis Encryption for Windows](#)
  5. For more information about Travis encryption using Linux and TravisCI see [Travis Encrypting](#).
5. After setup, Refer to [How to push build to Git using Appveyor and Travis](#)

[Working Example deploys/uploads to Cloudsmith and GitHub Release](#)  
[GitHub Issues discussion here Deploy to your repository is simple... #77](#)

## CIRCLECI (Build Account)

1. Open a (free) open-source account on circleci.com [Signup](#) with your GitHub account
2. You will be allowing CircleCI certain types of access to your GitHub account.
3. Start following your GitHub plugin project on Circleci.
4. In CircleCI
  1. Under settings create a "context" with [username]
  2. Prerequisites: Config.yml uses workflows and you must be an admin.
  3. Under settings "Enable Github Checks"  
<https://circleci.com/docs/2.0/enable-checks/#section=projects>
  4. Under Settings > VCS > "Manage Github Checks"? Yes
  5. [Environment Settings and API tokens](#)
  6. Context- Contexts provide a mechanism for securing and sharing environment variables across projects. The environment variables are defined as name/value pairs and are injected at runtime. <https://circleci.com/docs/2.0/contexts/>
5. In **Project > Build Settings > Build Settings > Environmental Variables** to set up security.
6. See [CLOUDSMITH\\_API\\_KEY Generation and Use](#)
7. Add environment variables to the job. You can add sensitive data (e.g. API keys) here, Private environment variables enable you to store secrets safely even when your project is public, see Building Open Source Projects for associated settings information. Use Contexts to further restrict access to environment variables from within the build, refer to the "Restricting a Context" documentation.
8. [Building Open Source Projects](#)
9. Put your **Cloudsmith Deployment Key** in your **Circleci account> Settings> [user]> squiddio\_pi> Env Variables** as "CLOUDSMITH\_API\_KEY" .
10. The squiddio\_pi/ci/circleci-upload.sh file uses the name "CLOUDSMITH\_API\_KEY"
11. Typical Settings: Since building a PR is one way to make sure it works, if someone wants to help you out, therefor please enable the *Build forked pull requests* option which should make circleci build all PRs. If *Pass secrets to builds from forked pull requests*, is Disabled, it will not be able to upload anything.



# CLOUDSMITH Deployment Repository

Create your own CLOUDSMITH Deployment Repository. The shipped configuration uses cloudsmith.io. This is not a requirement in any way, but this is how to get the shipped setup working. Setup a Cloudsmith.IO Free Opensource Account:

1. Go to Cloudsmith.io <https://cloudsmith.io/user/signup/> to sign up for a free opensource account using your github account.
2. After the 7 day trial, you may be asked for credit card, as long as you stay public opensource with the base services, it should not cost you. Don't get alarmed when it looks like they want money. Open-source repos are free.
3. Create [user name] For Example: "firstname-lastname"
4. Create 3 repositories named
  1. [pluginname\_pi]-prod
  2. [pluginname\_pi]-beta
  3. [pluginname\_pi]-alpha
5. OR (if you plan on deploying multiple plugins)
  1. opencpn-plugins-prod
  2. opencpn-plugins-beta
  3. opencpn-plugins-alpha
6. Which will store compressed binary files and metadata xml catalog built by circleci.
7. Create a [new repository] by clicking the "+" and then use one of the names above. Make sure you fill in the 'Slug' with exactly the same text as you used in 'Name' or you repository may not work as you expect and debugging is difficult.
8. Make sure you select "Opensource" (not Public) and complete the rest of the information appropriately, see below.

The screenshot shows the 'CREATE PACKAGE REPOSITORY' form. Red arrows point from the 'Open-Source' radio button to the 'Open-Source License' dropdown and the 'Open-Source Project URL' text field. A red circle highlights the three 'I agree' checkboxes.

**CREATE PACKAGE REPOSITORY**

**Repository Owner Account** ⓘ

user: Iskillen

**Name** ⓘ

MyRepo

**Slug** ⓘ (optional)

my-repo

**Description** ⓘ (optional)

You don't need to add a description - We'll generate a short one for you.

**Repository Type** ⓘ

☐ Public ⓘ

☐ Private ⓘ

☒ Open-Source ⓘ

**Open-Source License** ⓘ

MIT License [mit] (OSI)

**Open-Source Project URL** ⓘ

https://github.com/some-account/some-repository

☒ I agree that the project and dependencies are free and/or open-source [by definition](#) ⓘ.

☒ I agree to be polite and appropriately attribute that Cloudsmith is providing free hosting.

☒ I agree that I have significant control and/or responsibility for the project.

1. When completed the new repository should have a "heart" icon which when hovered over shows "Opensource".
2. For each new repository select:
  1. Under "Set me up" dropdown on the right side, select "Raw"
  2. See notes below for setting up [CLOUDSMITH\\_API\\_KEY Generation and Use](#).
  3. Test each repository to make sure you can upload and download a file using the web UI.
  4. Then at the bottom "Upload Packages" button.

## CLOUDSMITH\_API\_KEY Generation and Use

KEY #2 provides your BUILD service the ability to PUSH artifacts (tarballs, xml, packages) to your CLOUDSMITH repository.

Provide your unique CLOUDSMITH\_API\_KEY to other services such as CIRCLECI, TRAVIS and APPVEYOR to give them permission to DEPLOY to CLOUDSMITH.

1. Copy the API key from your CLOUDSMITH Account on <https://cloudsmith.io> by
  1. Get the key from cloudsmith account page. Click the user name icon, upper right, and you will be able to copy the key to clipboard.
  2. Left click your logon-id in the top right of the cloudsmith main page
  3. Then click 'API Key'
  4. You will be asked for 'Elevated Privileges' and need to supply your password.
  5. Then your key will be displayed.
  6. Copy this (either select the whole key and copy or click the paper clip icon at the right hand end of the key line)
2. Now use your Cloudsmith key in the appropriate accounts below to provide access to Cloudsmith for deployment.
  1. Sign into your CIRCLECI account on <https://circleci.com>
    - Select 'Settings' in the left hand vertical toolbar
    - Then 'Projects'
    - Then the settings icon (gear wheel) of the project (plugin\_pi) you want to use.
    - Click 'Environment Variables' under 'BUILD SETTINGS',
    - Click "Add Variable"
    - Paste the Cloudsmith key into the Value Field..
    - Add 'CLOUDSMITH\_API\_KEY' into the Name Field.
    - Close by clicking Add Variable.
  2. Sign into your APPVEYOR account on [appveyor.com https://appveyor.com](https://appveyor.com)
    - First add the plugin project. The plugin name should show in the upper left.
    - From the plugin page go to Settings in the top menu.
    - Select Environment from the side menu.
    - Under Environment Variables select Add a Variable.
    - Paste the Cloudsmith key into the Variable Field and enter CLOUDSMITH\_API\_KEY into the Name Field.
    - Click Add Variable.
  3. Sign into your TRAVIS account on <https://travis-ci.org>
    - Select the plugin project so that it shows at the top.(eg: weather\_routing\_pi)
    - Select from the upper right "More Options". Then "Settings" from the dropdown.
    - Scroll down to Environment Variables.
    - Paste the Cloudsmith key into the Value Field and enter CLOUDSMITH\_API\_KEY into the Name Field.
    - I select All Branches.
    - Select Add. See the screenshot.

## Cloudsmith Retention Policy Settings

Some more information on how retention works here: <https://help.cloudsmith.io/docs/retention-lifecycle>

## Cloudsmith Repositories Examples

<https://cloudsmith.io/~david-register/repos/>

- <https://cloudsmith.io/~alec-leamas/repos/>

- <https://cloudsmith.io/~rick-gleason/repos/>



- [rick-gleason/opencpn-plugins-beta](#)
- [rick-gleason/opencpn-plugins-pkg](#)
- [rick-gleason/opencpn-plugins-prod](#)

<https://cloudsmith.io/~jon-gough/repos/>

- [jon-gough/testplugin\\_pi-beta](#)
- [jon-gough/testplugin\\_pi-pkg](#)
- [jon-gough/testplugin\\_pi-prod](#)

# Setup: Configure Plugin for PI Installer

## Summary of Process

1. Configure your plugin's files by replacing cmake and circleci scripts and modify them appropriately.
2. Create free opensource accounts with circleci and cloudsmith, and set up opensource repositories.
3. Test building and deployment to the cloudsmith "beta" repository.
4. Use Opencpn with the PI Installer to open the new [ocpn-plugin.xml metadata catalog to confirm installation and test the plugin.
5. After testing, issue a new version, pushing new binaries to cloudsmith repositories, along with a new ocpn-plugin.xml catalog file.
6. See [PI Installer Procedure Build-Deploy](#) below, for more detail.

## Add CMake and Script Files



Your plugin will need to be augmented with new CMake and script files. There are two versions you can select from Frontend1 and Frontend2.

1. The main Frontend1 example is in [github.com/bdbcat/oesenc\\_pi](#) and a more annotated version is in [rgleason/squiddio\\_pi branch frontend1](#)
2. The main Frontend2 example is in [github.com/jongough/testplugin\\_pi](#) and more annotated [asgithub.com/rgleason/squiddio\\_pi Branch frontend2](#)

The simplest way is to copy the new files in ci & cmake to your system and make the requisite changes to the CMakeLists.txt, of which there shouldn't be too many as it is really the individual plugins file, i.e. has all the customization in.

All changes for customisation should be constrained to the CMakeLists.txt file, because all the other files are parameter driven and so 'should' be the same between plugins. So all the files in the:

1. 'cmake' directory, including the in-files directory, are the same for every plugin, there should be no

customisation to these files. There are 'extra' cmake files which a plugin may use, i.e. FindPortaudio.cmake for the weather\_routing\_pi, but these are supplied by the plugin and are referenced via CMakeLists.txt in the plugin customisation section.

2. 'ci' directory should not require hand customisation as again all the files are parameter (global variable) driven
3. '.circleci' directory should not require hand customisation as again all the files are parameter (global variable) driven
4. 'debian' directory should not require hand customisation
5. 'mingw' directory should not require hand customisation

Updates of plugins require copying the above directories in place and then carefully updating the CMakeLists.txt file by referencing the testplugin\_pi version to change it to the new format and include all the 'standard' parts that are needed. It should be quite easy to get it working, while testing on the web takes longer as jobs need to finish, to determine what needs fixing.

So if you look through the first section of the CMakeLists.txt you will see where you set the cloudsmith user and repository name as well as the 'special' stuff for the xml file. The next section down you may need to change a few default settings, i.e. 'USE\_GL', or some special version of c++ that is needed. In the current file the following section (line 194 onwards) is where you define all the files to be used. You will need to keep 'SRCS' as the source list, but the rest of it is up to you. You will also need the last section which does the rest of the build and package process.

I would not try to 'combine' this process with any other in the same 'stream' or you are likely to have problems. When I make changes just copy the new files in place and, if needed, make the co-requisite changes to CMakeLists.txt .

The idea of this process is that it is a 'black box' to most and it should 'just work'. You will notice that I have changed the names of the repositories to '...-prod', '...-beta' and '...-alpha', it just seemed to match what was going into them. The destination repository is controlled by what you are doing, i.e. -

```
Any non-master branch network build -> alpha repository
Master build without tag and non-master branch build with tag -> beta
repository
Master build with tag -> prod repository
```

All 'installation' files, 'deb', 'dmg', 'exe', etc will also go into the same repository, but they will have the current naming strategy, i.e. will start with 'opencpn-plugin-' the the rest of the descriptive name.

Download and Use the CMake and CI files listed below from Jon Gough's [Testplugin\\_pi](#) using the "cmake\_flatpak\_test" branch. Download and install the files into a new branch on your plugin local repository.

Files

-----

```
CMakeLists.txt  <----- Your version and this version will have to be merged.
appveyor.yml
travis.yml
```

## Directories and Files

```
-----
cmake
circleci
buildosx
ci
debian
mingw
api-16
data (icons)
```

## Modify Files

Then the top of CMakeLists.txt must be customized for the plugin and environment. There are basically two sets of files for CMake that have been developed. One set comes from Alex Leamas and the other set comes from Jon Gough's testplugin\_pi which is being used as a template. Jon has gone to some effort to bring all the Plugin Dev settings up to the top of CMakeLists.txt The example below is from Jon's testplugin\_pi but needs to be updated!

### CMakeLists.txt

```
project(testplugin_pi)

set(PACKAGE_NAME testplugin_pi)
set(VERBOSE_NAME testplugin)
set(TITLE_NAME testplugin)
set(CPACK_PACKAGE_CONTACT "Jon Gough")

set(VERSION_MAJOR "1")    <---Set your version number and comment
set(VERSION_MINOR "0")
set(VERSION_PATCH "41")
set(VERSION_TWEAK "8")
set(VERSION_DATE "03/12/2019")
set(OCPN_MIN_VERSION "ov50")
set(OCPN_API_VERSION_MAJOR "1")
set(OCPN_API_VERSION_MINOR "16")
set(TP_COMMENT " * Release for 05 using CI")

set(PARENT "opencpn")
set(PACKAGE "testplugin")    <--- Set plugin name (twice)
```

```
set(VERBOSE_NAME "Testplugin")
#set(GIT_USER "jongough")      <--- Git user commented out
set(GIT_USER "rgleason")      <--- Set your git user name
set(GIT_REPOSITORY_NAME "testplugin_pi") <---Set the Git Repository Name
message(STATUS "CIRCLECI: ${CIRCLECLI}, Env CIRCLECI: $ENV{CIRCLECI}")
if($ENV{CIRCLECI})
    set(GIT_REPOSITORY_HOST "github.com")
    set(GIT_REPOSITORY_DIR "${GIT_USER}/")
#    set(GIT_REPOSITORY_BRANCH "master")
    set(GIT_REPOSITORY_BRANCH "cmake_flatpak_test")
else()
    set(GIT_REPOSITORY_HOST "git.eclipse.com.au") <--- This is used if you
    setup your own git server environment for testing
    set(GIT_REPOSITORY_DIR "")
    set(GIT_REPOSITORY_BRANCH "cmake_flatpak_test")
endif()
set(CLOUDSMITH_USER "rick-gleason") <--- Set your Cloudsmith
Name
#set(CLOUDSMITH_BASE_REPOSITORY "${GIT_REPOSITORY_NAME}") <---Uses git repos
name for Cloudsmith repositories
set(CLOUDSMITH_BASE_REPOSITORY "opencpn-plugins") <---Uses a generalized
repository for multiple plugins.
set(XML_INFO_URL "https://opencpn.org/OpenCPN/plugins/ocpn_draw.html")
set(XML_SUMMARY "Test of ODraw ODAPI and JSON interfaces")
set(XML_DESCRIPTION "Test ODraw API and demo use from another plugin")
set(APPVEYOR_TEST_DEPLOY_TO_CLOUDSMITH "true")
```

## circleci\config.yml

Your account does not have rights to run macos until CircleCI staff explicitly enables it. If you don't have access to run the macOS environment that particular build will not start but all the others should run OK. **Therefore comment out the build for “macos” lines, until you ask for and are given permissions** by them to run limited use macos. **First you need to create a real userid login in addition to your github login to get MacOS.** See Build on macOS<https://circleci.com/open-source/>**Then you need to write Circleci Support** to [billing@circleci.com](mailto:billing@circleci.com) including your “Opensource” account, stating that you are creating only public OpenSource for OpenCPN and would like MacOS build capability.

Also write staff nicely asking if you can get permission for MacOS, and stating that the plugin is opensource for OpenCPN.

```
Line 39 start
#    build-macos:
```

```
#   macos:
#     xcode: "10.0.0"
#     environment:
#       - OCPN_TARGET: macos
#     steps:
#       - checkout
#       - run: ci/circleci-build-macos.sh
#       - run: ci/circleci-upload.sh
```

```
Line 65 start
#     - build-macos:
#       filters:
#         branches:
#           ignore:
#             - devel
#             - tmp
```

## appveyor.yml

Generate a GitHub Personal Token specifically for Appveyor and your plugin. Save the code somewhere. Then encrypt the Github Personal Token with appveyor's encryption, and put that encryption into the code as below.

```
deploy:
  provider: release    # or GitHub
  auth_token:          # git-ptoken-squiddio-into-appveyor [whatever you name it in
                        # GitHub]
  secure: "<encryption from appveyor https://ci.appveyor.com/tools/encrypt>"
```

## .travis.yml

Generate a GitHub Personal Token specifically for Travis and your plugin. Save the code somewhere. Then encrypt the Github Personal Token with Travis CI, and put that encryption into the code as below.

Refer to [Travis Encryption of Github Personnal Access Token](#)

```
deploy:
- provider: releases
  api_key:    # git-ptoken-squiddio-into-travis [whatever you name it in
              # GitHub]
  secure: <add TravisCI encryption of git ptoken key>
  repo: [username]/squiddio_pi #Deployment to GitHub Release Tag when a tag
is pushed.
  tags: true
```

all\_branches: true

## Other Notes

### Cloudsmith OpenCPN Organization

<https://cloudsmith.io/orgs/opencpn/>

There are 3 repositories that can be used.

Developers of Plugins can be invited to join the organization after they have created an opensource Cloudsmith account.

See this <https://github.com/OpenCPN/OpenCPN/issues/1573>

### Installation Destination change with Plugin Manager

<https://github.com/OpenCPN/OpenCPN/issues/1605>

Check your plugins for any changes needed to keep intended user experience when installing, noting path issues.

### Catalog XML "name" must match "Common Name"

The name used in the Plugin Manager XML Catalog must match the "Common Name" of the plugin being installed.

In order for the Plugin Manager to work consistently, the "name" tag in the catalog's XML must match the "Common Name" of the plugin being installed. The (sometimes inconsistent) values in CmakeLists.txt do not affect the plugin manager at run-time. Example catalog xml record:

```
<plugin version="1">
  <name> WeatherRouting </name>
  <version> 1.13.2 </version>
  <release> 0 </release>
  <summary> Plugin to complete optimal routing with weather files </summary>
  <api-version> 1.16 </api-version>
  <open-source> yes </open-source>
  <author> Sean d'Epagnier </author>
  <source> https://github.com/rgleason/weather_routing_pi </source>
```

```
<description> Weather_Routing Plugin optimizes weather routing</description>
<target>msvc</target>
<target-version>10.0.14393</target-version>
<tarball-url>
https://dl.cloudsmith.io/public/rick-gleason/opencpn-plugins-beta/raw/names/we
ather_routing-msvc-10.0.14393-
tarball/versions/1.13.2.+80.c284e85/weather_routing_pi-1.13.2.0-
ov50-1.16_msvc-10.0.14393.tar.gz</tarball-url>
<info-url> https://opencpn.org/OpenCPN/plugins/weather_routing.html </info-
url>
</plugin>
```

The name "WeatherRouting" must match the "Common Name" which comes from the source code, weather\_routing\_pi.cpp, in github:

```
wxString weather_routing_pi::GetCommonName()
{
    return _("WeatherRouting");
}
```

From:

<https://opencpn.org/wiki/dokuwiki/> - **OpenCPN Manuals**

Permanent link:

[https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer\\_manual:pi\\_installer\\_build\\_deploy](https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer_manual:pi_installer_build_deploy)

Last update: **2020/06/22 12:12**

